rebar3 Erlang build tool

START HERE

`rebar3` **`new app`** *`<name>`*
Create an OTP app

`rebar3` **`ct`**
Run Common Test suites

`rebar3` **`compile`**
Fetch deps and build

`rebar3` **`dialyzer`**
Type analysis

`rebar3` **`shell`**
Erlang shell with the project

`rebar3` **`as prod tar`**
Release tarball

`rebar3` **`eunit`**
Run unit tests

`rebar3` **`help`** *`<task>`*
Help for a task

rebar.config deps, profiles → `compile` → **rebar.lock** pinned deps → `build` → **_build/<profile>** beams, deps, rel → `release / tar` → **Release** relx, start script

CONCEPTS

rebar3 is the Erlang build tool: one rebar.config per project, builds land in _build/<profile>. Run rebar3 help to list tasks, including plugin ones.

Profiles pick the build variant: default, test (used by eunit and ct) and prod. Select with rebar3 as <profile> <task>.

Commit rebar.lock so every machine builds the same dependency versions.

PROJECTS

`rebar3` **`new app`** *`<name>`*
OTP application with supervisor

`rebar3` **`new lib`** *`<name>`*
Library without a supervisor

`rebar3` **`new release`** *`<name>`*
Umbrella project for releases
umbrella is an alias of the release template; apps go under apps/.

`rebar3` **`new escript`** *`<name>`*
Command-line escript project

`rebar3` **`new help`**
List available templates

BUILD & RUN

`rebar3` **`compile`**
Fetch deps and compile all apps

`rebar3` **`clean`**
Remove compiled beams

`rebar3` **`clean --all`**
Clean deps and all profiles too

`rebar3` **`shell`**
Shell with apps and deps loaded
Starts the apps in the shell; use --start-clean to load code without starting them.

`rebar3` **`shell --sname`** *`<name>`*
Shell as a named node

`rebar3` **`shell --apps`** *`<a>,<b>`*
Start only these apps

`rebar3` **`shell --config`** *`<file>`*
Use a sys.config file

`rebar3` **`do`** *`<task>,<task>`*
Run several tasks in a row

`rebar3` **`escriptize`**
Package an escript executable

`rebar3` **`version`**
Show rebar3 and Erlang versions

TEST

`rebar3` **`eunit`**
Run all EUnit tests

`rebar3` **`eunit --module=`***`<mod>`*
Tests of one module

`rebar3` **`eunit --test=`***`<mod>:<fun>`*
Run one test function

`rebar3` **`eunit --application=`***`<app>`*
Tests of one app

`rebar3` **`eunit --file=`***`<path>`*
Tests from a source file

`rebar3` **`eunit -v`**
Verbose output

`rebar3` **`ct`**
Run all Common Test suites
*Suites are *_SUITE.erl files in each app's test/ directory.*

`rebar3` **`ct --suite=`***`<path>`*
Run one suite
Comma-separate several suites; the path has no .erl extension.

`rebar3` **`ct --suite=`***`<path>`* **`--group=`***`<g>`*
Run one test group

`rebar3` **`ct --suite=`***`<path>`* **`--case=`***`<c>`*
Run one test case

`rebar3` **`ct --sys_config=`***`<file>`*
Use a sys.config for tests

`rebar3` **`ct --verbose`**
Print more output

`rebar3` **`eunit --cover`**
Collect coverage data

`rebar3` **`ct --cover`**
Collect coverage data

`rebar3` **`cover --verbose`**
Coverage report in the console

`rebar3` **`do eunit,ct`**
Run both test types

DEPENDENCIES

`rebar3` **`deps`**
List deps and lock status

`rebar3` **`tree`**
Show the dependency tree

`rebar3` **`upgrade`**
Upgrade all dependencies

`rebar3` **`upgrade`** *`<dep>`*
Upgrade one dependency
Also upgrades the subtree of that dependency.

`rebar3` **`unlock`** *`<dep>`*
Drop a dependency from the lock

`rebar3` **`update`**
Refresh the Hex package index

DEPENDENCIES (cont.)

deps in rebar.config

```
{deps, [
  cowboy,
  {jsx, "~> 3.1"},
  {mylib, {git, "<url>", {tag,
"1.0.0"}}},
  {other, {git, "<url>",
{branch, "main"}}}
]}.
```

Work on a dependency locally by putting a copy or symlink in _checkouts/. The deps entry must still exist.

List every dependency in the .app.src applications list too, or it will not start with your app.

PROFILES

`rebar3` **`as`** *`<profile> <task>`*
Run a task in a profile

`rebar3` **`as`** *`<p1>,<p2> <task>`*
Combine profiles

`rebar3` **`as test compile`**
Build with test profile deps

profiles in rebar.config

```
{profiles, [
  {test, [{deps, [meck]}]},
  {prod, [{erl_opts,
[no_debug_info]}]}
]}.
```

QUALITY

`rebar3` **`dialyzer`**
Build the PLT and check types
The first run builds a PLT and is slow; later runs reuse it.

`rebar3` **`xref`**
Cross-reference check

`rebar3` **`edoc`**
Generate API docs

Checks in rebar.config

```
{dialyzer, [{warnings,
[no_return]}]}.
{xref_checks,
[undefined_function_calls,
 exports_not_used]}.
{cover_enabled, true}.
```

RELEASES

`rebar3` **`release`**
Build a dev release

`rebar3` **`as prod release`**
Release with ERTS, no symlinks

`rebar3` **`as prod tar`**
Release as a .tar.gz

`_build/`*`<profile>`***`/rel/`***`<name>`***`/bin/`***`<name>`** **`console`**
Start a release in a shell

`bin/`*`<name>`** **`foreground`**
Run in foreground (systemd, Docker)

RELEASES (cont.)

`bin/`*`<name>`** **`daemon`**
Run as a background node

`bin/`*`<name>`** **`remote_console`**
Attach to a running node

`bin/`*`<name>`** **`ping / stop / restart`**
Control a running node

relx in rebar.config

```
{relx, [
  {release, {<name>, "0.1.0"},
[<name>, sasl]},
  {dev_mode, true},
  {include_erts, false},
  {extended_start_script, true}
]}.
```

CONFIG & PLUGINS

Compile options

```
{erl_opts, [debug_info,
 warnings_as_errors]}.
{minimum_otp_vsn, "26"}.
```

Aliases and hooks

```
{alias, [{check, [xref,
dialyzer, eunit, ct]}]}.
{provider_hooks, [{pre,
[{compile, clean}]}]}.
```

Plugins

```
{plugins, [rebar3_auto]}.
{project_plugins, [erlfmt]}.
```

Plugins for every project go in ~/.config/rebar3/rebar.config. project_plugins are used only when building this project directly.

HEX

`rebar3` **`hex user auth`**
Log in to hex.pm

`rebar3` **`hex search`** *`<term>`*
Search packages

`rebar3` **`hex publish`**
Publish package and docs

RECIPES

New app, test, run

```
rebar3 new app myapp && cd
myapp
rebar3 eunit
rebar3 shell
# 1> observer:start().
```

CI check

```
rebar3 do xref, dialyzer,
eunit, ct
# or define an alias and run:
rebar3 check
```

Production tarball

```
rebar3 as prod tar
tar xzf _build/prod/rel/<name>/
<name>-*.tar.gz -C /opt/<name>
/opt/<name>/bin/<name>
foreground
```

Debug one failing test

```
rebar3 ct --suite=test/
<name>_SUITE --case=<case>
rebar3 shell --eval
"<mod>:<fun>()."
```